Simulation of Advanced Driver System for a Merging Situation in Highway with MATLAB/Simulink.

Graz University of Technology

Institut für Regelungs- und Automatisierungstechnik

Roger Gustems Maestre

Supervisors:

Ass.Prof. Dipl.-Ing. Dr. techn. Martin Steinberger

Dipl.-Ing. BSc Astrid Rupp

Graz, April 2018

Table of Contents

1 Introduction 5
2 Problem statement 6
3 Car model 8
3.1 Car model using Simulink 8
3.2 Controller design 8
3.2.1 String stability 10
4 Overview of the code implementation 13
4.1 Car model using MATLAB code 14
4.1.1 Data structure 15
5 Simulation Scenario and Final Approach Description 17
5.1 Zone 1: The free zone 17
5.2 Zone 2: The control zone 18
5.2.1 Decision-making for merging sequence 18
5.2.2 Positioning 19
5.2.2.1 Polynomial generation for cruise velocity 20
5.2.2.2 Polynomial generation for distance reduction 21
5.2.2.3 Maximum acceleration 23
5.2.2.4 Change between controllers 27
5.3 Zone 3: The changing-lane zone 27
6 Implementation Progress and Code Evolution 28
6.1 Video function 31
7 Results 37
8 Conclusions and Future Work 39
9 References 40

Index of Figures

Figure 1 :Lane reduction on a 2-lane highway..... 6
Figure 2: Scheme of the closed loop of a car 10
Figure 3: Acceleration using PD controller with feed-forward and constant time-headway. 11
Figure 4: Velocity using PD controller with feed-forward and constant time-headway. 11
Figure 5: Distance between cars using PD controller with feed-forward and constant time-headway. 11
Figure 6: Structure of the code in MATLAB 13
Figure 7: Schematic of the data structure 15
Figure 8: New order after platoon fuse 18
Figure 9: Merging two platoons into one 18
Figure 10: Order of merging process of two platoons 19
Figure 11: Trajectory planning of a 4th order polynomial 21
Figure 12: Trajectory planning of a 5th order polynomial 23
Figure 13: Velocity graphic of 5-car platoon with fixed maximum acceleration 24
Figure 14:Acceleration graphic of 5-car platoon with fixed maximum acceleration 24
Figure 15: Velocity graphic of 5-car platoon with waiting and positioning flags 25
Figure 16: Acceleration graphic of a 5-car platoon with waiting and positioning flags 25
Figure 17: Velocity graphic of a 5-car platoon with variable maximum acceleration 26
Figure 18: Acceleration graphic of a 5-car platoon with variable maximum acceleration 26
Figure 19: Changing lane process for cars in second lane 27
Figure 20: Acceleration using PD controller with feed-forward and constant time-headway with large acceleration on the leader 28
Figure 21: Velocity using PD controller with feed-forward and constant time-headway with large acceleration on the leader 28
Figure 22: Position using PD controller with feed-forward and constant time-headway with large acceleration on the leader 29
Figure 23: Distance between cars using PD controller with feed-forward and constant time-headway with large acceleration on the leader 29
Figure 24: Platoon acceleration graphic on a first approach 30
Figure 25: Platoon velocity graphic on a first approach 30
Figure 26: Position plot with cars in both lanes 31
Figure 27: Position graphic of video fuse1.avi 32
Figure 28: Velocity graphic of video fuse1.avi 33
Figure 29: Acceleration graphic of video fuse1.avi 33
Figure 30: Position graphic of video fuse2.avi 34

Figure 31: Velocity graphic of video fuse2.avi 34
Figure 32: Acceleration graphic of video fuse2.avi 34
Figure 33: Schematic of the data structure in previous versions 36
Figure 34: Position plot of video final1.avi 37
Figure 35: Velocity plot of video final1.avi 38
Figure 36: Acceleration plot of video final1.avi 38

1 Introduction

In recent years, the capacity of highways has become a crucial factor in the creation of traffic jams. Due to the increase in traffic on the roads and the impossibility of increasing the capacity of the road, a difficult situation has been created that leads to an increase in traffic jams with significant economic losses. An easy way to increase the capacity of these roads is to decrease the distance between the cars while maintaining the same speed. This approach immediately raises the problem of safety and collisions between high-speed cars. For these reasons, it is very important to use ACC (adaptive cruise control). The ACC automatically adapts the distance and speed to the vehicle in front and can be used to reduce the impact of deficiencies in the reliability of the driver, for example, stability control, longitudinal control, sudden emergency brake, etc.

There is a possibility that in one situation, one or more lanes will be blocked on the highway. In this case, an approach is needed to merge the traffic flow with the remaining lanes. This maneuver is performed by human drivers every day, but in automated driving systems, it remains a challenge [15]. The purpose of this project is the implementation of a decentralized merging algorithm for a two-to-one lane reduction on highways. The system could reduce traffic congestion, particularly by reducing the distance between vehicles. According to [20], “Grouping vehicles into platoons is a method of increasing the capacity of roads. An automated highway system is a proposed technology for doing this. Platoons decrease the distances between cars or trucks using electronic, and possibly mechanical, coupling. This capability would allow many cars or trucks to accelerate or brake simultaneously”. Building and managing a platoon requires multiple technologies and poses problems in the topology of the network and communications [10] and [11] .To ensure the coordination of the vehicles, the following are essential: i) a control algorithm that regulates the relative distance with respect to the vehicle ahead and coordinates all the vehicles to stabilize the platoon, some examples of control design can be seen in [12], and ii) a communication network to exchange information among other platoons, some examples are can see in [13] These technologies also offer new opportunities for cooperative driving vehicles The control aim is the collision-free merging of two platoon of vehicles. The control algorithm can use data received from multiple vehicles in the platoon. In this work is assumed that each vehicle knows its own position and the position of the vehicle ahead. The vehicles are equipped with communication modules, which facilitate completely new strategies for the coordination of the vehicles.

2 Problem statement

This project has been divided in two different parts. The first part consists in creating car models and platoons. In the second part, the merging algorithm is implemented. In the first part, it has been considered a simple longitudinal vehicle model, composed of a double integrating system. Without any passive loss due to air drag or internal friction in the engine, transmission or wheels. For these reasons, the longitudinal dynamics of the car in the state space can be described as

$$\dot{Y}_i(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} Y_i(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_i(t) \tag{1}$$

where

$$Y_i(t) = \begin{bmatrix} x_i \\ v_i \end{bmatrix} \tag{2}$$

and x_i denotes the position, v_i the velocity and u_i the control input (acceleration) for the vehicle i.

The merging algorithm has been developed on a straight highway of two lanes (focusing only on one traffic direction), with a lane reduction due to some obstacles in the second lane (left lane) as shown in Figure 1. Before explaining the method that is used, it is important to take into account the continuity problem that this scenario raises. Going from two lanes to only one implies that the output capacity of the system is reduced by half while maintaining the same input capacity. Similar to the narrowing of pipes in a continuous flow in stationary state, the sum of the in-flows has to be the same as the out-flows in a time interval. If the output capacity is smaller than the input capacity, the particles in the output have to go faster than in the input, otherwise, there would be an accumulation within the system.

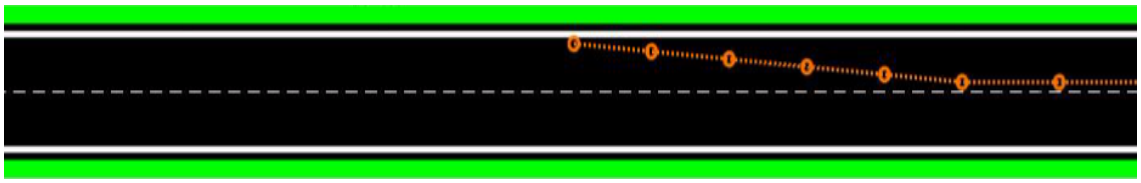

Figure 1 :Lane reduction on a 2-lane highway

Hence, the idea is that the vehicles increase their speed while the merging algorithm is executed. This ensures that the cars in the input will not reach the cars that are in the merging process.

The scenario shown in Figure 1 will be divided into three different zones. The first zone corresponds to the normal highway without any obstacle before the "decision-making point". The second zone (also called the "control zone") goes from the decision-making point to 300 meters before the merging point. In this zone is where the entire positioning algorithm will be executed. And the last zone

goes from the end of the second zone to the merging point, where all the cars will be in the same lane. In this small area, cars in the second lane will change to the first lane.

The approach for this method is based on several publications, extracting ideas from [15], [16], [17], [18], [19]. There are many possible approaches, in this case it has been decided to develop a cooperative merging system similar to the zipper principle. For this reason, the cars will be distributed in platoons on both lanes and run the merging algorithm to increase the efficiency of this procedure.

In addition, the creation of a decentralized system that allows merging cars on the highway without the need for a coordinator is one of the key points. The implementation of a coordinator in the merging system implies a great use of resources, due to the need of the coordinator to know all the information of all the cars and process all the information to be able to give orders to each car in the region. This process takes a long time or the use of a powerful processing unit, which is expensive.

With a decentralized approach, the cars execute the merging algorithm. Instead of a single central processor that analyzes all the information, each car can execute its own analysis of the road and adapt to the situation. Also, due to the simplicity of a decentralized system, only a signal is needed that indicates that the cars must execute the merging algorithm, this system can be placed in a small time anywhere on the road.

It is also assumed that all cars are equipped with an ideal communication system. Consequently, all the information shared by the cars is accurate (without communication errors) and without time-delay. The merging algorithm can use data received from multiple vehicles inside and outside the platoon. The main objective of the communications is to maintain the cohesion of the platoon.

To simplify the project and have a good overview of the problem, the following assumptions have been made: First, the flow of traffic entering the scenario is always constant. Second, the platoon leader does not make sudden brakes or big brakes. Third, all cars maintain a speed between 20 and 40 m/s.

3 Car model

As mentioned in the previous chapter, the model used for the cars is a double integrator system. The cars are represented by a point in space without mass and, consequently, without inertia. For this reason, only the cinematic aspect of the system is relevant. Also, they are not considered losses due to friction or drag forces. This statement leads directly to Newton's first law of motion [1] which says that without any input force to the system, it will continue with constant velocity and zero acceleration.

3.1 Car model using Simulink

Before starting, it has to be considered that since it is a system controlled by a computer or CPU, the control cannot be in continuous time. For that reason, discrete-time plants and controllers have been investigated and a sample time of 0.1 seconds has been chosen.

In a first approach, the Simulink block of state-space has been used, but soon was discarded. Two discrete-time integrator blocks [2] replaced the state-space block, this second approach provides a greater understanding and clarity to the system. These blocks also added features to the model such as easier access to the state variables of the system (position and velocity) or saturation in the output (it is undesired behavior for the car to have negative velocity or high module of the acceleration).

The discrete-time integrator block has an option to decide the integration method used for computing the next state. The method to compute the current state responds to the idea that, with the information of the previous state and the acceleration provided by the algorithm, the current state is obtained. For this reason, the backward Euler method (3) has been chosen as integration method.

$$y(k) = y(k-1) + \Delta t \cdot u(k) \tag{3}$$

where k is the current time step and Δt is the sample time of the system.

3.2 Controller design

Once the car model is defined, the focus can be laid on the controller and the closed loop. In order to control the vehicle i, a PD controller has been implemented. Also, a feedforward has been added to the control action. This feedforward is the acceleration of the vehicle in front $(i-1)$ and provides faster response to any action that the preceding car could do. As result, the control action can be expressed as

$$u_i(k) = k_p \cdot e_x + k_d \cdot e_v + u_{i-1} \tag{4}$$

and the position and velocity error can be computed as

$$\begin{cases} e_x = x_{i-1} - x_i - (Th \cdot v_i + d_{min}) \\ e_v = v_{i-1} - v_i \end{cases} \tag{5}$$

where x is the position, v is the velocity of the cars, d_{min} is a constant distance in meters and Th is the distance from the tip of one vehicle to the tip of the next one behind it, expressed as the time it will take for the follower vehicle to cover that distance. These parameters determine the distance with respect to the car that is being followed. The first parameter d_{min} is a constant that indicates the minimum distance between the cars when they have zero speed. The second parameter Th is called the time-headway that generates a relationship between the speed of the vehicle and the distance to maintain with the in front vehicle. The higher the velocity of the car, the longer the distance to the car in front, and vice versa. In this case, it is also a constant (in seconds) that is multiplied by the velocity resulting in a distance in meters.

For the design of the controller, the $lqrd$ function that MATLAB provides has been used. The $lqrd$ function calculates a state-feedback controller, the states of the system have to be modified. Instead of using the states in (1), the states will be the position error e_x and the velocity error e_v

$$\begin{cases} e_x = x_{i-1} - x_i - C \\ e_v = v_{i-1} - v_i \end{cases} \tag{6}$$

were C being a constant assuming the terms $(Th \cdot v_i + d_{min})$ are constant. Then, the system can be written

$$\dot{e} = \begin{bmatrix} \dot{e_x} \\ \dot{e_v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} e + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (u_{i-1} - u_i) \tag{7}$$

or

$$\begin{bmatrix} \dot{e_x} \\ \dot{e_v} \end{bmatrix} = \begin{bmatrix} e_v \\ u_{i-1} - u_i \end{bmatrix} \tag{8}$$

Taking u_i from (4)

$$\begin{aligned} \dot{e}_v &= u_{i-1} - u_i \\ &= u_{i-1} - u_{i-1} - k_p \cdot e_x - k_d \cdot e_v \\ &= -\begin{bmatrix} k_p & k_d \end{bmatrix} \cdot \begin{bmatrix} e_x \\ e_v \end{bmatrix} \end{aligned} \tag{9}$$

This leads to the conclusion that the controller obtained with the $lqrd$ function is suitable as a PD controller for the system.

$$[\mathrm{K}, \mathrm{S}, \mathrm{e}] = \mathrm{lqrd}\,(\mathrm{A}, \mathrm{B}, \mathrm{Q}, \mathrm{R}, \mathrm{Ts}) \tag{10}$$

where A and B are the matrixes in (7), that are the same matrixes that appear in (1), Ts is the sample time and Q and R are the weight matrixes for the states and the control signals respectively.

The weight for the velocity error is bigger than the position error in the Q matrix, in order to prevent larger accelerations due to big position errors. Even if the position error is big, if the relative velocities of the cars are small, the collision will not occur. The R matrix has been set to a large value to penalize large control inputs (accelerations). The $lqrd$ function designs a discrete-time state-feedback controller that has similar characteristics to a continuous state-feedback controller designed using the lqr function. The $lqrd$ function calculates the controller using the state-feedback law

$$u(n) = -K \cdot Y(n) \tag{11}$$

with

$$K = [k_p \quad k_d] \tag{12}$$

that minimizes a discrete-time cost function equivalent to the continuous cost function

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{13}$$

As a result, the scheme referring to a single vehicle seen in Figure 2 can be obtained.

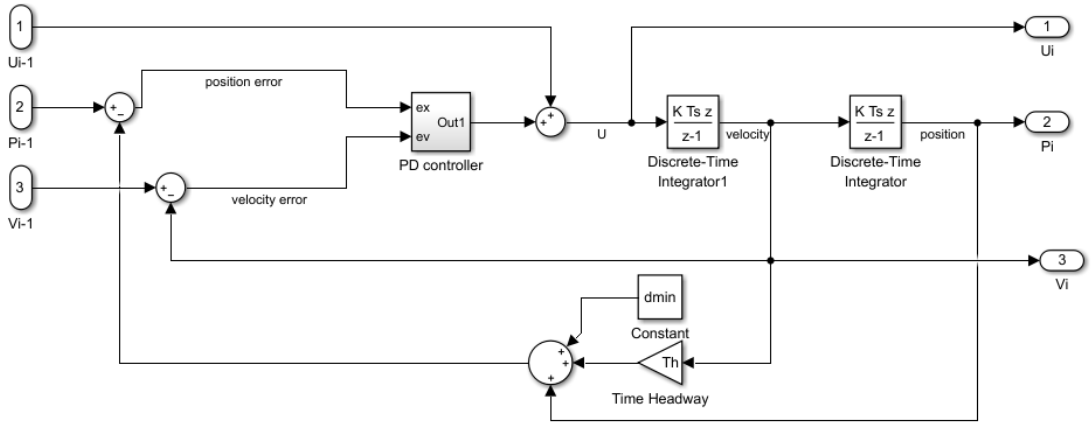

Figure 2: Scheme of the closed loop of a car

3.2.1 String stability

In a platoon, multiple vehicles in the same lane follow the first vehicle called “leader” and have to modify their velocity to keep distance with the predecessor vehicle. The acceleration of the leader can be arbitrary and the “followers” have to adapt their behavior to the car in front. Although the PD controller proposed in the last section provides stability to the car, it is not sufficient to guarantee string stability. String stability has been defined in different ways [4], [5], [6], [7] and [9]. In this work, the string stability will be concluded when the platoon is collision-free and there is no amplification of the accelerations through the platoon.

For this purpose, a constant time-headway (Th in (5)) and feed forward policy has been chosen. As seen in (4) where the control action has already the acceleration of the predecessor vehicle and in (5) where the position error has the constant distance and the constant time-headway. This approach ensures the string stability for this project. As discussed in [4], the feed forward ensures the collision-free platooning with perfect communications. Also, the fact of adding the constant time-headway spacing helps to smooth the accelerations over the string. The results of using these two policies is shown in Figures 3,4 and 5.

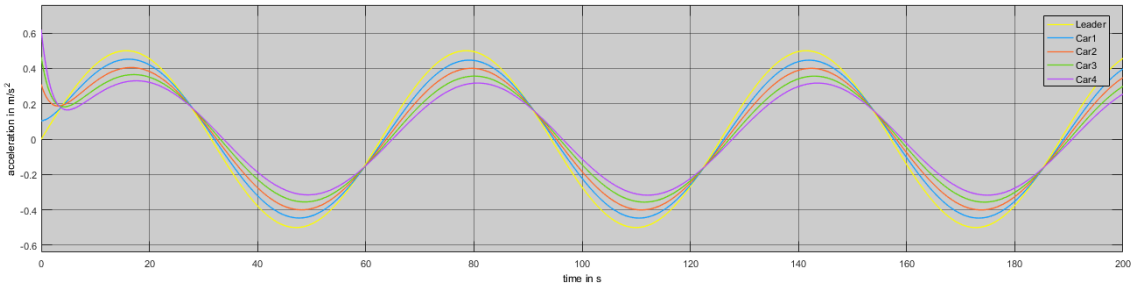

Figure 3: Acceleration using PD controller with feed-forward and constant time-headway.

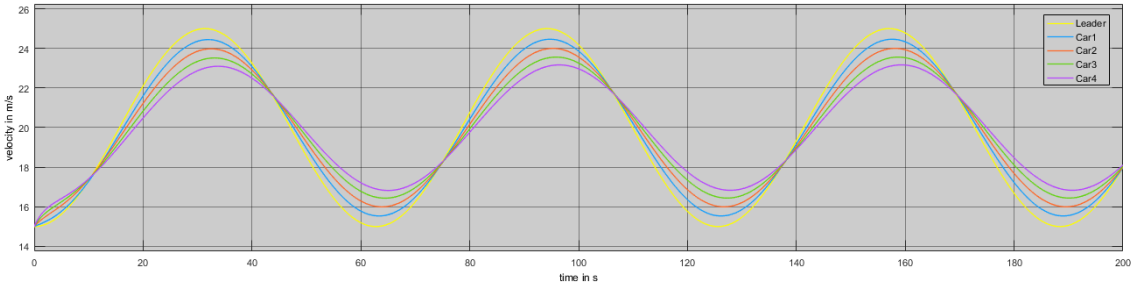

Figure 4: Velocity using PD controller with feed-forward and constant time-headway.

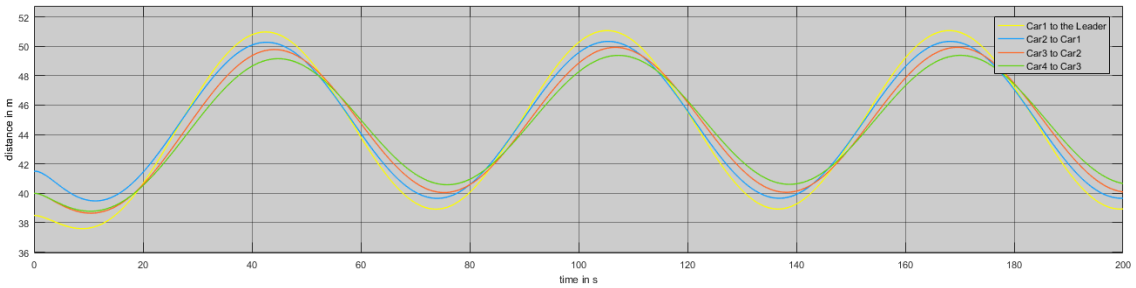

Figure 5: Distance between cars using PD controller with feed-forward and constant time-headway.

As can be seen in Figure 3, the leader car has been fed with a sinusoidal acceleration of amplitude 0,5m/s^2 in this experiment. In Figure 3 and 4 can be seen the attenuation of the acceleration and velocity of the followers as expected. Also, Figure 5 shows that the distance between the cars is never lower than zero (this indicates that there are no collisions between cars). Then, with these results it can be affirmed that the control fulfills all the requirements for string stability and ensures a collision-free driving.

Next, for each simulation a random n number of cars in the platoon has been considered instead of working only with a fixed number of cars. Simulating arbitrary platoon length raises the problem of tedious work in Simulink. Also, the complexity of the model started to be a key factor that leaded to change the Simulink model to MATLAB code. This change provides benefits and simplicity to the project, but also has disadvantages.

4 Overview of the code implementation

The code implementation follows the structure shown in Figure 6. At the beginning, there is the initialization of the variables and the controller. After the initialization, the time steps loop starts. Every time step the possibility of adding new cars to the scenario is checked. Also, in every simulation step, there is a loop for all the cars in the scenario.

```
for i=0:simulation_steps
    % displaying simulation step
    disp(i)

    % add trafic to the simulation
    […] = add_traffic(…);

    % proper loop starts here
    cars_ids=fieldnames(cars);
    for k=1:length(cars_ids)
        cid=char(cars_ids(k)); % pointer using dynamic reference

          ⋮ % all code here

    end % loop for cars
end %loop simulation
```

This second loop does not have the same number of cars for all times. Due to the inflow and outflow of cars, the cars that enter the scenario are added to the struct and the cars that exit the scenario after the merging point are removed and stored in another struct to reduce the load on the processor. After the simulation ends, all the cars are saved in the same struct for the analysis of the simulation and the plots. Once in the car loop, the dynamic reference feature that the structs have makes it easier to work with the data structure and access to its features.

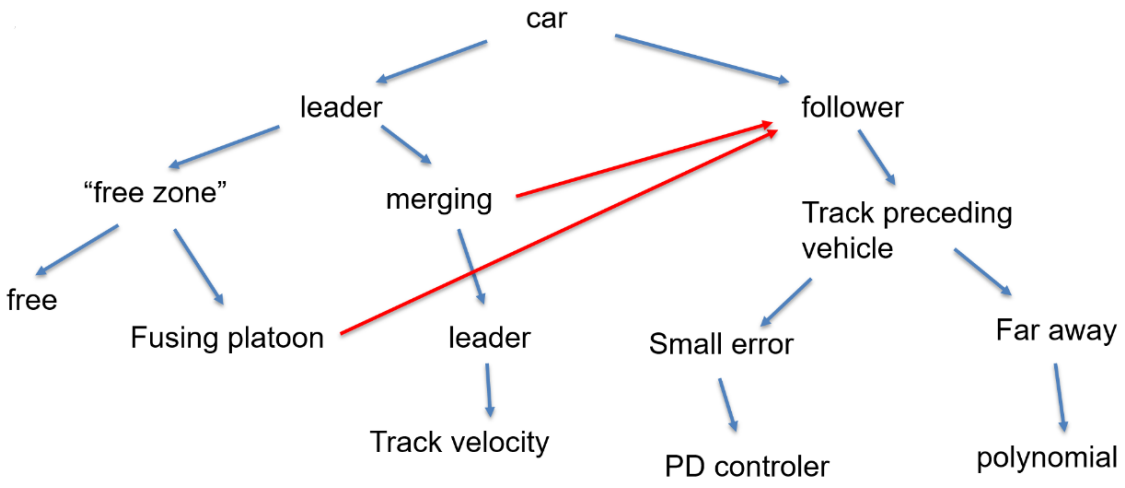

Figure 6: Structure of the code in MATLAB

First for each car in the loop, the position is checked in case it is needed to update the zone in which it is. The zones will influence the behavior as will be explained in the next chapter. After this update, the main difference between the cars appears. There is a difference in the treatment of the leaders and the followers.

For this reason, the code is split into two branches the way to calculate the control action of the vehicles (acceleration). The first branch is for the leader cars. In case it is in the first zone, the car will have random acceleration or, if the platoon goes to close with the platoon in front, they will fuse. In this case, the leader of the platoon will become a follower the next simulation step and take the other branch. In case of crossing the decision-making point two things could happen. The first thing is that in the new platoon, the car keeps being the leader. In that case, it will be the responsible to reach the cruise velocity. In the other case, the leader will become a follower in the new platoon.

The second branch is for all the other cars that are followers. The follower behavior is always the same, follow the leader with a certain distance. This distance depends on the zone the car is. For the follower cars, the position error determines which approach is used to accomplish the desired distance. In case of a small position error, the cars use the PD controller. In the other case, the cars will have a large position error and will use a trajectory planning approach to reduce the velocity as explained in the next chapter.

4.1 Car model using MATLAB code

The change from Simulink model to MATLAB code was straightforward with the equations of the backward Euler method (3). A function has been created that computes the current state with the previous state and the control action as input variables. This function also includes the saturation for the velocity and acceleration that the discrete-time integrator block allowed. Once the function is executed, it returns the value of position, velocity and acceleration of the current time step.

```
function [new_p,new_v,u]=oneStep(x,v,u,T)
%function to update all parameters --> 2integrator

% saturation of the acceleration
if u>8
    u=8;
elseif u<-8
    u=-8;
end

% integrator "forward euler" velocity
new_v=v+T*u;

% saturation velocity
if new_v<0
    new_v=0;
    u=0;
elseif new_v>40
    new_v=40;
    u=0;
end

% integrator "forward euler" position
new_p=x+T*new_v;

end
```

The main drawback with the MATLAB code is the storage of information for each car and each time step. Due to the amount of information generated for each car, it cannot be stored in simple matrixes. The solution to this problem is explained in the next Section.

4.1.1 Data structure

For this project, the structure of the data has been a very important challenge. Although it is not in the control field, it acquires large importance when the simulation runs on MATLAB code and not in Simulink.

Every car has been defined as a MATLAB struct object. This car object has different properties, flags, variables and a matrix that stores all the information of each simulation step. The stored values can be accessed by the fieldname, which provides robustness to the code implementation. To create the main loop for all the cars in the simulation each time step, another struct has been created. This global struct has as the fieldname the car id (id#) and it is linked to the car struct. In summary, there is a big struct that stores all the car structs. The structure can be seen in Figure 7.

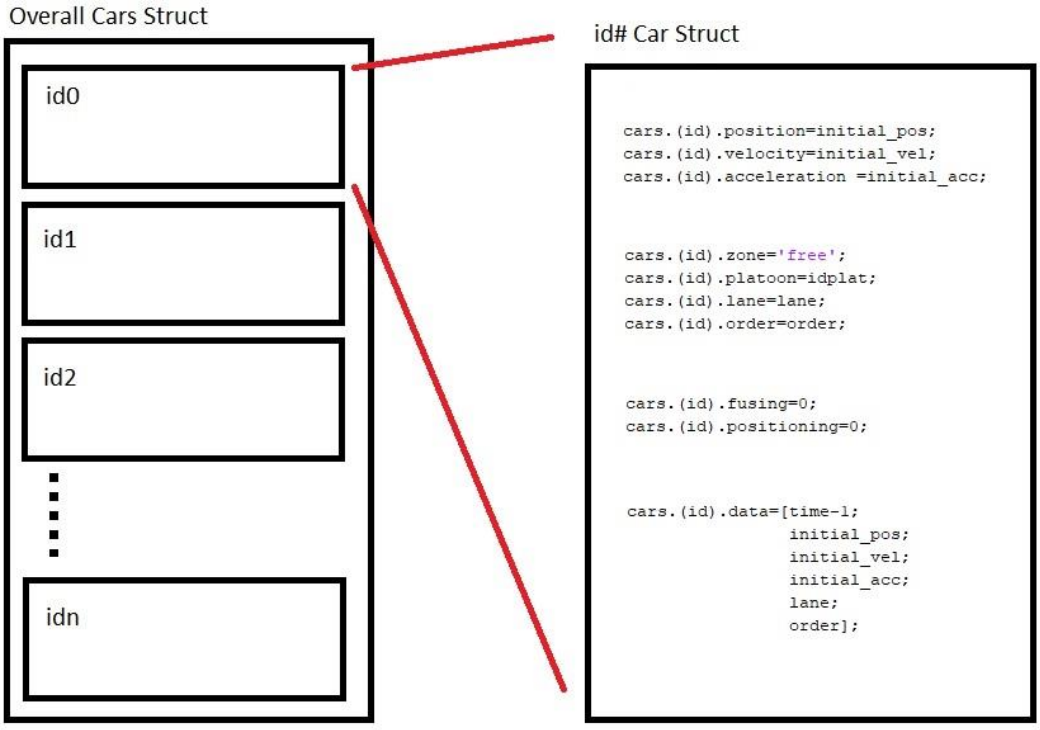

Figure 7: Schematic of the data structure

Also, the struct object has been chosen because they provide the “dynamic field reference” feature. This feature helps to access each of the cars using a variable (as the car id) and select the car needed for each step or function. This is very important because each car has a different ID. Moreover, this data structure allows having all the information stored in one variable.

To distinguish all the cars in the different platoons, every car has a field called *platoon* (as can be seen in Figure 7). All the cars of the same platoon have the same value un this field. In addition, in the same platoon, the cars are ordered by the value in the field *order*. The leader will have order = 1, the second car will have order = 2 and so on until the last car.

5 Simulation Scenario and Final Approach Description

In section 3.2, the implementation of a suitable PD controller that guarantees a collision free platoon has been explained. As mentioned in chapter 2, the distance between the cars has to be reduced and the velocity increased to avoid traffic jams in the merging point. This controller is important in the following steps because the cars will be driving with high velocity with small distance between them. In this chapter is explained how is divided the highway in the previous meters to the merging point and the methods that will be used in each zone. Also, the final approach implemented for each of these zones is explained.

5.1 Zone 1: The free zone

The first zone is called the “free zone” in this project. This zone corresponds to the normal highway without any obstacles before the “decision-making point”. This decision point is the position on the road where the cars have to decide, using the decision-making algorithm, which position they will take in the final platoon. As the name indicates, in this first zone the merging process is not applied, but it is necessary to simulate the behavior of the platoons on a highway. The cars introduced in the simulation in both lanes at the beginning of this zone represents the car flow distribution in an average two-lane highway. The cars are initialized and introduced in the scenario in platoons of n cars on both lanes with a certain velocity. This initial velocity follows a normal distribution centered in 30 m/s (108 km/h).

Remark: The macroscopic traffic analysis is not part of this work; some ideas are based on [8]. It is not explained why these values have been taken, only the numerical values that has been implemented.

Due to the possible velocity difference between two platoons of the same lane, the second platoon (with higher velocity) may reach the first one. In this case, a function that allows to fuse the two platoons into one has been implemented. Once the leader of the second platoon is close enough, the function changes the “platoon” attribute of the car structs to the first platoon, and the order of the cars according to the number of cars on the first platoon. E.g. if the car in the second platoon has order 2 and the first platoon has 3 cars, the new order will be 2+3=5. This procedure raises the problem of a large position and velocity error from one step to the next, because the leader of the second platoon will be a follower the next time step, see Figure 8. To avoid the large acceleration provided for the PD controller, the “positioning” flag of the leader of the second platoon is set to one and a trajectory planning is calculated to reduce the distance. The trajectory

planning used here is the same used for reducing the gap between cars in the merging sequence and will be explained in the next section.

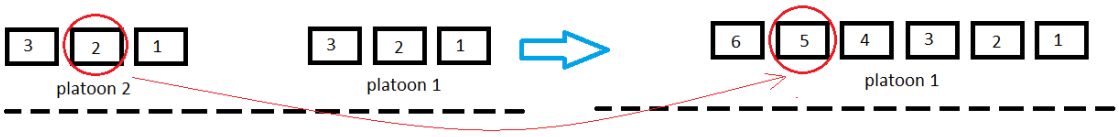

Figure 8: New order after platoon fuse

5.2 Zone 2: The control zone

At the beginning of the zone 2 (also called the “control zone”), the decision-making algorithm is executed, and the cars are positioned according to the new order in the overall platoon (see Figure 9). This zone goes from the decision point to 300 meters before the merging point, where the zone 3 starts and the cars have to change to the first lane.

5.2.1 Decision-making for merging sequence

Once the cars in the second lane have entered the control zone, the order that they will have in the new platoon has to be decided. For this procedure an algorithm called the decision-making algorithm is used.

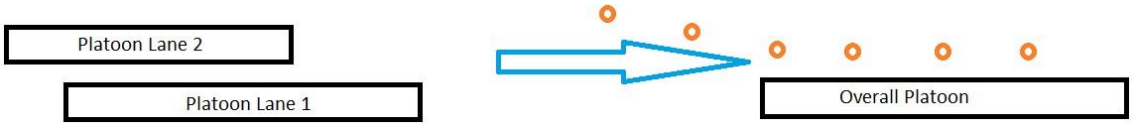

Figure 9: Merging two platoons into one

The algorithm scans the surroundings of the car to find the suitable candidates to be the predecessor and the follower. The scan area is set from 200 meters in front of the car (in both lanes) to 200 meters behind the car, but only in the first lane. The aim of this procedure is to find the order in the overall platoon by accommodating the cars from the second lane to a platoon of the first lane. The reasons for choosing the lanes for the scan is because all the cars in front of the car that is making the decision have already taking that decision and they know their order in the overall platoon (in the first lane), but the cars behind in the second lane still have to make that decision. When a car in the second lane arrives at the decision point, it is assumed that all the cars in front have already taken the decision (in the overall platoon), but not the cars behind.

Remark: if platoon 1 is (AAAAA) and platoon 2 is (BBBBB), the new order it is not necessary (ABABABABAB). See example in Figure 10 that the new order is (AABABBABAB)

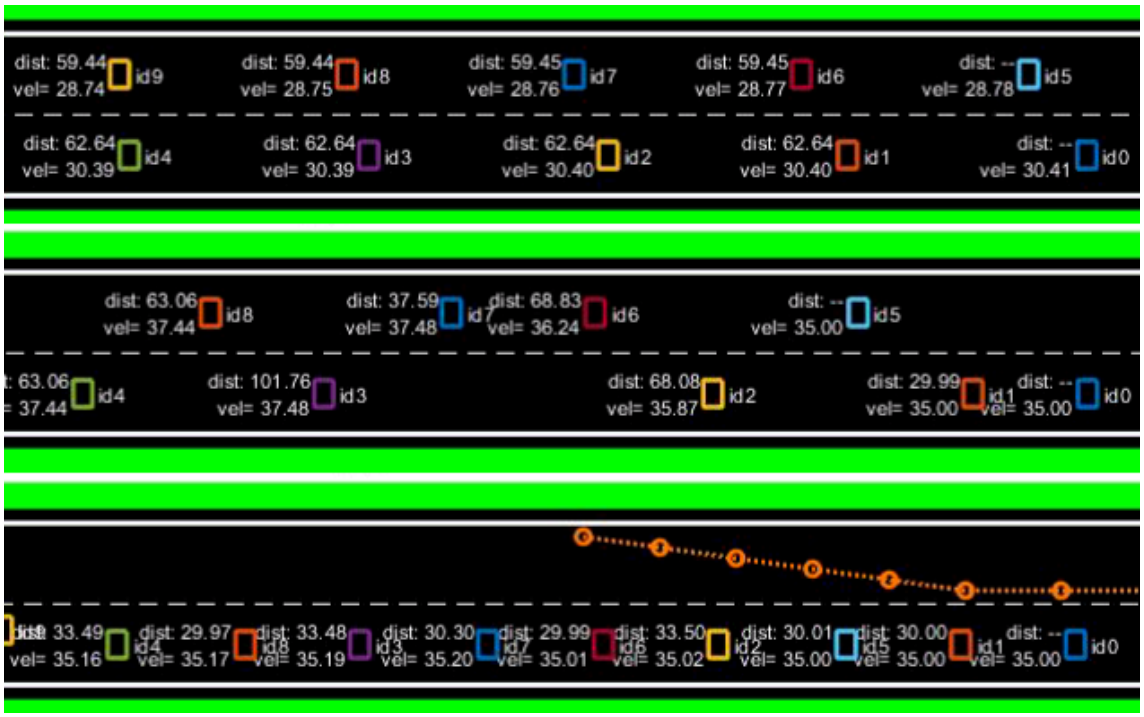

Figure 10: Order of merging process of two platoons

After finding all the candidates to be the predecessor and the following car, the difference of position and velocity is computed using

$$f_i = (x_0 - x_i) + k \cdot (v_0 - v_i) \tag{14}$$

Where x_i are the positions and v_i the velocities of car i, and the sub index 0 indicates the reference car and i are the other detected cars. The constant k is a numerical value in seconds to relate the difference of position and velocity. With all the values obtained in (10), only the lowest positive value and the highest negative value are taken in consideration. If

$$f_i > 0 \tag{15}$$

the car i will merge behind the considered car. Following the same rule, the car i will be in front of the car 0, if

$$f_i < 0 \tag{16}$$

5.2.2 Positioning

After the decision point, all the cars have to adapt the velocity and reduce the distance with respect to the preceding car. Also, after running the decision-making algorithm, there will be only one leader for the two platoons (the overall platoon). This leader is in charge to increase the velocity of all the platoon to the merging velocity. The other leader is now a follower and has to modify the position and velocity according to the platoon.

5.2.2.1 Polynomial generation for cruise velocity

As mentioned in previous chapters, the velocity of the cars when they leave the scenario is an important factor in this merging system. The velocity of the cars has to increase, and the distance has to be reduced in order to maintain the input-output flow in the control region.

For this reason, the leader of the platoon has to plan the increment of velocity. The cruise velocity that has been chosen is 35m/s (126km/h). This velocity is slightly higher than the average entrance velocity (30m/s or 108km/h). This approach and the reduction of the distance between cars is sufficient to maintain the continuous flow of vehicles in the given examples, despite the lane reduction. However, the distance reduction will be explained in another section.

This procedure is based on the trajectory planning of [4]. The jerk is the third derivate of the position and is the most important variable for to create a smooth trajectory for the comfort of the passengers. The trajectory is computed in order to minimize the cost function

$$J = \int_{t_0}^{t_f} \frac{1}{2} j(t)^2 dt \tag{17}$$

for the initial time $t_0 = 0$ to a final time t_f and $j(t)$ is the longitudinal jerk. Since there is no condition for the final position, this coefficient can be freely chosen, and in consequence the order of the polynomial is four.

Remark: if there is a constraint for the end position, it is recommended to use the fifth order polynomial that will be described in the next section.

After solving the optimization problem stated as

$$\begin{aligned} &\min_{j(t)} \int_0^{t_f} \frac{1}{2} j(t)^2 dt \\ &\dot{x}(t) = v(t) \\ &\dot{v}(t) = a(t) \\ &\dot{a}(t) = j(t) \end{aligned} \tag{18}$$

with

$$\begin{aligned} x(0) = x_0 \,;\, v(0) = v_0 \,;\, a(0) = a_0 \\ v(t_f) = v_f \,;\, a(t_f) = 0 \end{aligned} \tag{19}$$

The longitudinal states can be expressed as

$$j(t) = C_1 t - C_2 \tag{20}$$

$$x(t) = \frac{1}{24}C_1 t^4 - \frac{1}{6}C_2 t^3 + \frac{1}{2}C_3 t^2 + C_4 t + C_5$$

and the coefficients are defined by the initial and final states

$$\begin{gathered} C_5 = x_0 \; ; \; C_4 = v_0 \; ; \; C_3 = a_0 \\ C_2 = \frac{4a_0 t_f + 6(v_0 - v_f)}{{t_f}^2} \\ C_1 = \frac{6a_0 t_f + 12(v_0 - v_f)}{{t_f}^3} \end{gathered} \tag{21}$$

with

$$t_f = \frac{-3(v_0 - v_f)}{2a_{max}} \tag{22}$$

Then, with a_{max} chosen to provide comfort to the maneuver, the longitudinal polynomial for the trajectory planning can be found for the acceleration, velocity and position. The results are shown in the Figure 11.

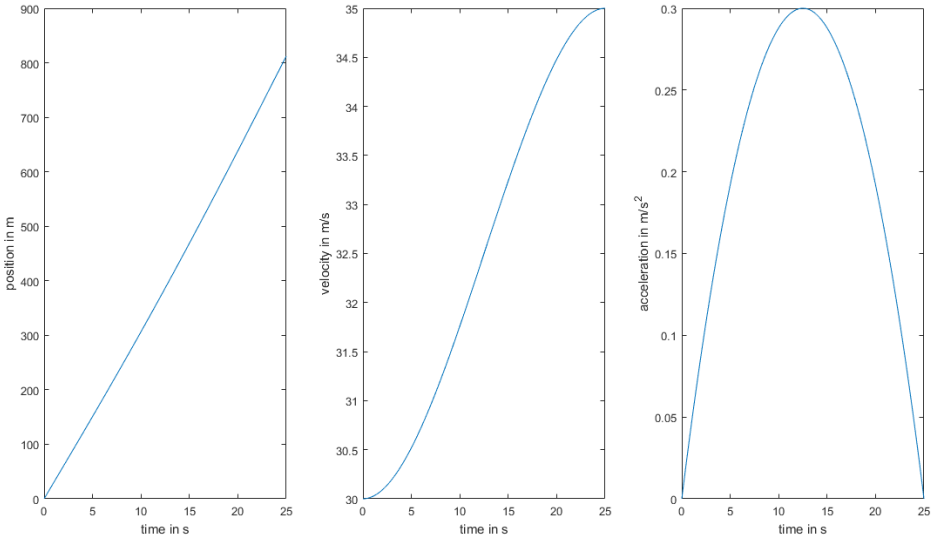

Figure 11: Trajectory planning of a 4th order polynomial

The trajectory planning function is executed every time step by the leader. The trajectory is calculated until the leader reaches a certain velocity error with the cruise velocity $e_v < e_{min}$. Then, it changes the control to the PD controller to keep the cruise velocity. The e_{min} value is set to ensure a smooth change between controllers.

5.2.2.2 Polynomial generation for distance reduction

As mentioned in the previous section, when there is a final position constraint, the use of fifth order polynomial is recommended. In the case of large position or

velocity error, or if the distance requirements change suddenly due to fusing platoons or reducing gap between cars, the PD controller is not suitable. The appearance of a sudden peak in the error leads to large instantaneous accelerations to compensate the error. For these cases, the “positioning” flag is enabled and set to one, so the car can change from the PD controller to the polynomial generation. The polynomial generation for distance reduction follows the same principle as the cruise velocity, but the final position is fixed to avoid collisions.

This method is similar to (17) described in the previous section and further explained in [4]. The difference lies in the number of restrictions for the final state, incrementing the order of the polynomial. Then, the new set of constraints are

$$\begin{gathered} x(0) = x_0 \; ; \; v(0) = v_0 \; ; \; a(0) = a_0 \\ x(t_f) = x_{i-1} - d = x_f \; ; \; v(t_f) = v_{i-1} \; ; \; a(t_f) = a_{i-1} \end{gathered} \tag{23}$$

Where d is the desired distance between the cars. As can be seen, the final state is linked to the predecessor car. The polynomials for the trajectory planning are

$$\begin{aligned} j(t) &= -\frac{1}{2}C_1 t^2 + C_2 t - C_3 \\ a(t) &= -\frac{1}{6}C_1 t^3 + \frac{1}{2}C_2 t^2 - C_3 t + C_4 \\ v(t) &= -\frac{1}{24}C_1 t^4 + \frac{1}{6}C_2 t^3 - \frac{1}{2}C_3 t^2 + C_4 t + C_5 \\ x(t) &= -\frac{1}{120}C_1 t^5 + \frac{1}{24}C_2 t^4 - \frac{1}{6}C_3 t^3 + \frac{1}{2}C_4 t^2 + C_5 t + C_6 \end{aligned} \tag{24}$$

and the coefficients are defined for the initial and final conditions

$$\begin{gathered} C_6 = x_0 \; ; \; C_5 = v_0 \; ; \; C_4 = a_0 \\ C_3 = \frac{3(3a_0 - a_{i-1}){t_f}^2 + 12(3v_0 + 2v_{i-1})t_f + 60(x_0 - x_f)}{{t_f}^3} \\ C_2 = \frac{12(3a_0 - 2a_{i-1}){t_f}^2 + 24(8v_0 + 7v_{i-1})t_f + 360(x_0 - x_f)}{{t_f}^4} \\ C_1 = \frac{60(a_0 - a_{i-1}){t_f}^2 + 360(v_0 + v_{i-1})t_f + 720(x_0 - x_f)}{{t_f}^5} \end{gathered} \tag{25}$$

Note that for the computation of the coefficients, the value of t_f is needed. This value is also linked to the a_{max} and has been calculated according to the procedure in [4]

$$t_f = \sqrt{\frac{10}{\sqrt{3}}\frac{1}{a_{max}}\left|x_f - x_0\right|} \tag{26}$$

This polynomial, like the cruise velocity, has to be calculated every time step. The reason is that the car and its predecessor are changing every time step the position, velocity and acceleration. This implies that the calculated polynomials cannot ensure the correct behavior for the car in the following time steps.

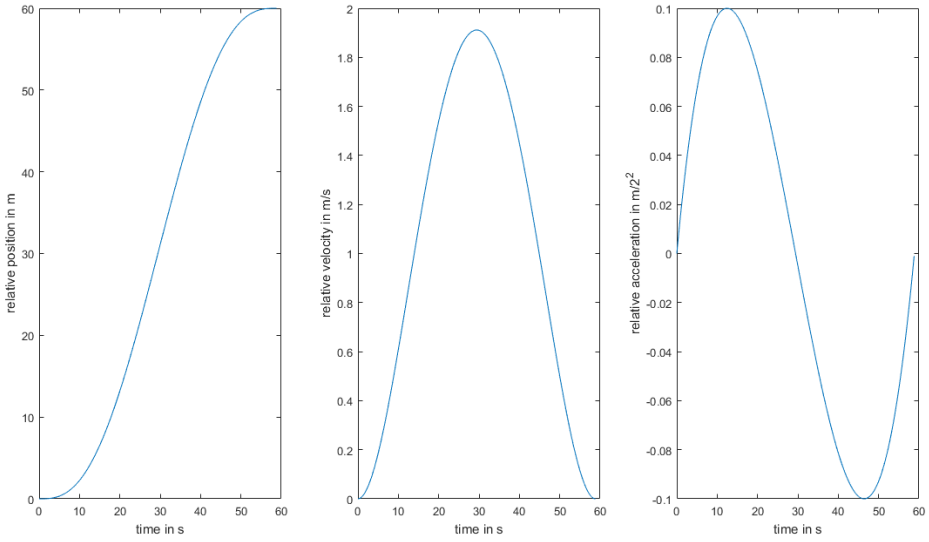

Figure 12: Trajectory planning of a 5th order polynomial

In Figure 12 the distance reduction can be observed, while the start and end velocities and accelerations are the same. The difference can be seen in curve of the acceleration, where there is an acceleration phase at the beginning to reduce the distance and after a brake phase to reduce the velocity according to the predecessor vehicle. Also, it is important to note that the initial and final velocities and accelerations are reduced to the ones of predecessor vehicle and the position reduced to the desired distance.

5.2.2.3 Maximum acceleration

The trajectory planning implemented for reducing the distance between cars depends on the parameter of the maximum acceleration. However, the cars may have too high accelerations due to the feed-forward. When the cars calculate the trajectory, the acceleration from the polynomial is added to the acceleration of the car in front because of the feed-forward. For this reason, if all the cars in the platoon are reducing the distance with fixed maximum acceleration at the same time, the second car has to reduce x meters with the car in front, but the third car has to reduce also x meters with respect the second car, this is $2x$ meters with respect the first car. In case on the nth car of the platoon, it has to reduce $(n-1)x$ meters with a total acceleration of $(n-1)a_{max}$. As can be seen in Figure 13, the higher the order of the car in the platoon, the higher the acceleration to reduce the distance. This behavior is highly undesired and dangerous for the passengers. In some cases, this values of acceleration and brake will exceed the mechanical limitations of the cars.

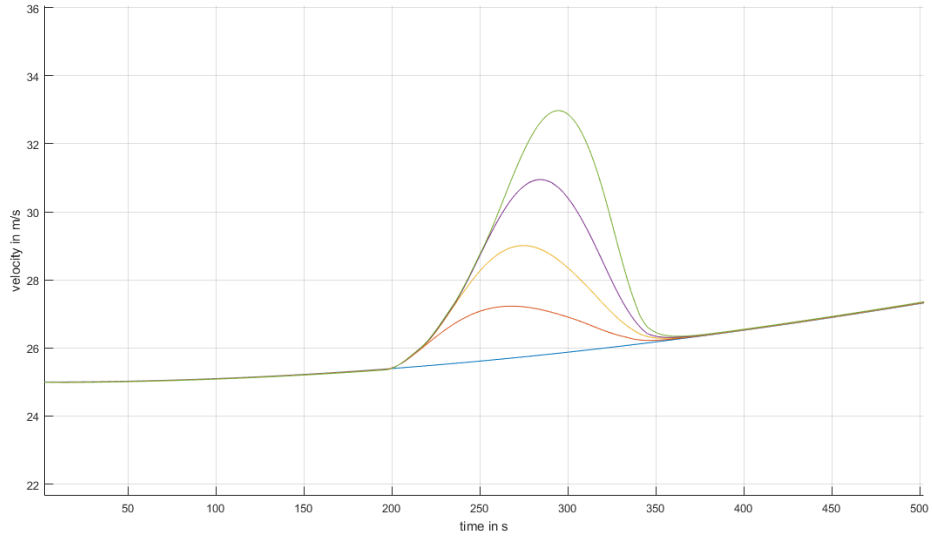

Figure 13: Velocity graphic of a 5-car platoon with fixed maximum acceleration, leader (blue), follower1(red), follower2(yellow), follower3(purple), follower4(green)

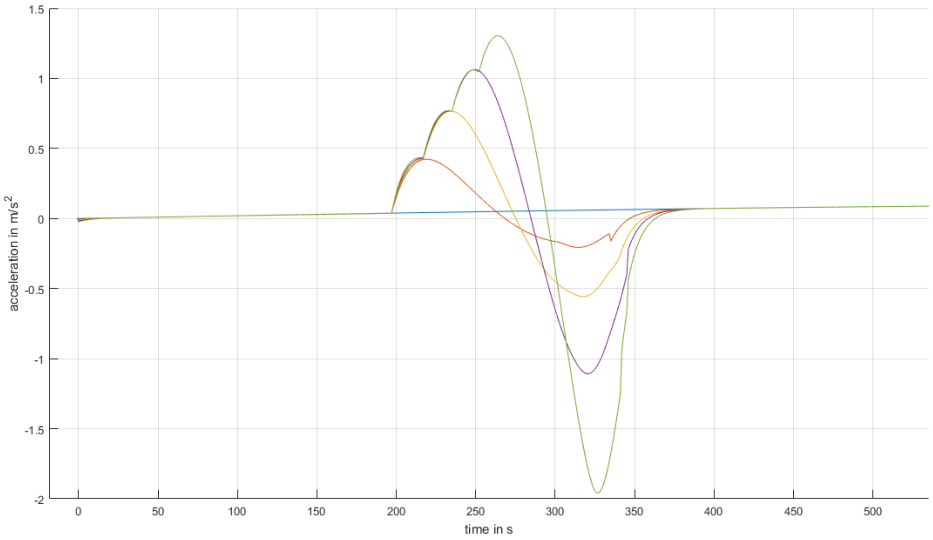

Figure 14:Acceleration graphic of a 5-car platoon with fixed maximum acceleration, leader (blue), follower1(red), follower2(yellow), follower3(purple), follower4(green)

For this reason, a flag feature was added. The cars had a flag called “waiting” flag, this flag was in charge of executing the distance reduction depending on what the predecessor car is doing. As explained before, when a car enters in the control zone, they have to reduce the distance with the car in front and the “positioning” and “waiting” flag are set to one for all the platoon except the first car that has “waiting” flag equals to zero. It means that the first car can do the maneuver while his followers maintain the long-distance platooning. Once the first car has completed the maneuver, the “positioning” flag was set to zero and changes from the trajectory planning to the PD controller because it has reached the desire velocity and distance with respect the car in front. At the same time,

the “waiting” flag of the second car is set to zero and allows the car to execute the trajectory planning. This method is applied until all the cars have reduced the distance between them. As can be shown in the Figure 15, this second approach raises the problem of low efficiency. When a car has to accelerate to increase the velocity needs consume energy (gas in combustion engines or electricity in electric cars), but when it reaches the final position it has to brake to adapt the velocity to the predecessor car. At this point, the car is dissipating all the energy used to increase the velocity. It is not a problem if the following cars don’t have to accelerate and brake again consuming and dissipating more energy. In conclusion, this approach is comfortable and safe, but energetically unacceptable.

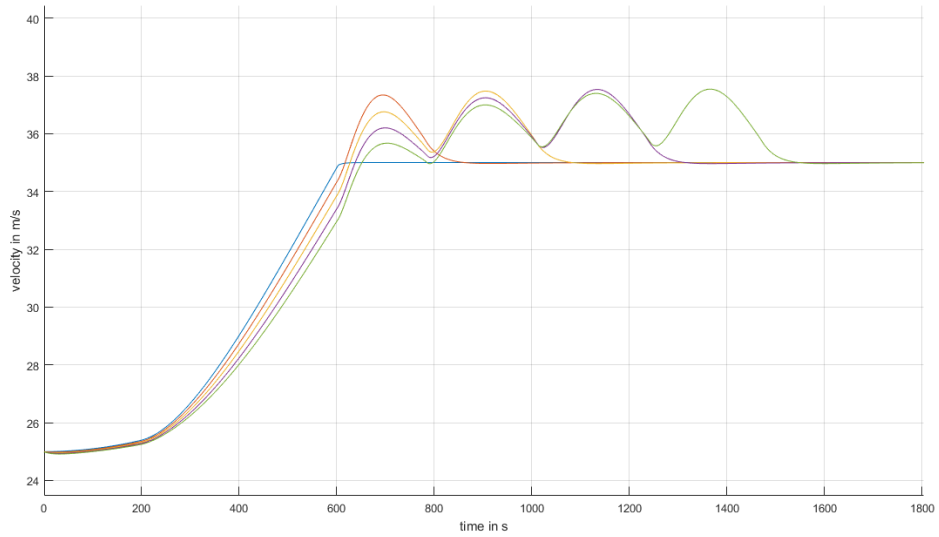

Figure 15: Velocity graphic of a 5-car platoon with waiting and positioning flags, leader (blue), follower1(red), follower2(yellow), follower3(purple), follower4(green)

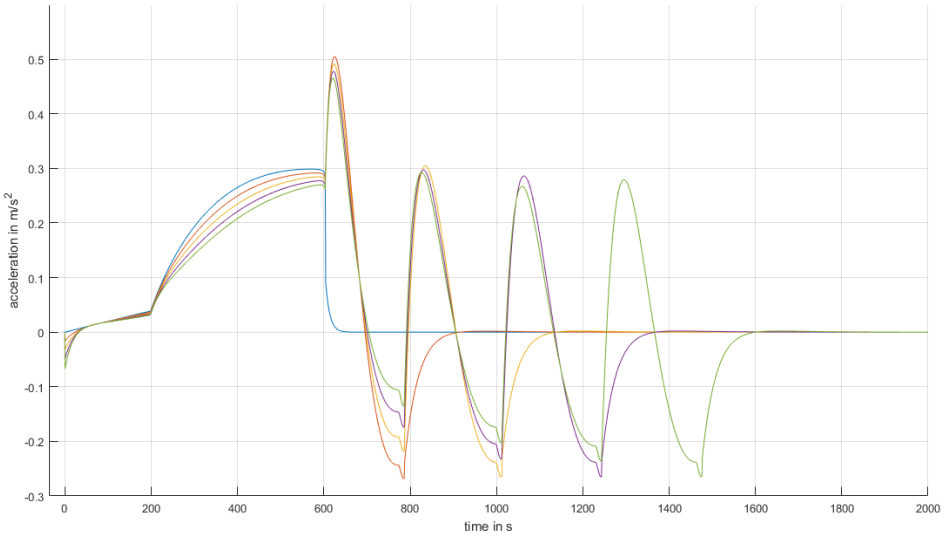

Figure 16: Acceleration graphic of a 5-car platoon with waiting and positioning flags, leader (blue), follower1(red), follower2(yellow), follower3(purple), follower4(green)

Finally, a third solution was implemented, which is a trade-off between both approaches. All cars will reduce the distance at the same time but with maximum total acceleration. The key of this approach is the use of the acceleration of the predecessor car to set the maximum acceleration

$$a_{\max i} = a_{max} - a_{i-1} \tag{27}$$

As can be observed in Figure 18, all the cars are executing the maneuver at same time but with bounded maximum acceleration. Also, the fact that the cars only accelerate and brake only once and the low accelerations have a big impact on the fuel consumption.

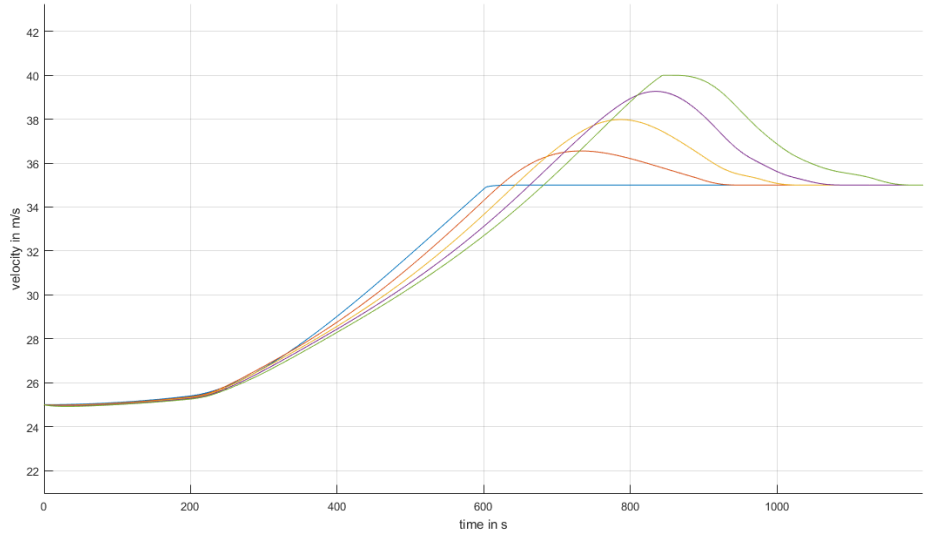

Figure 17: Velocity graphic of a 5-car platoon with variable maximum acceleration, leader (blue), follower1(red), follower2(yellow), follower3(purple), follower4(green)

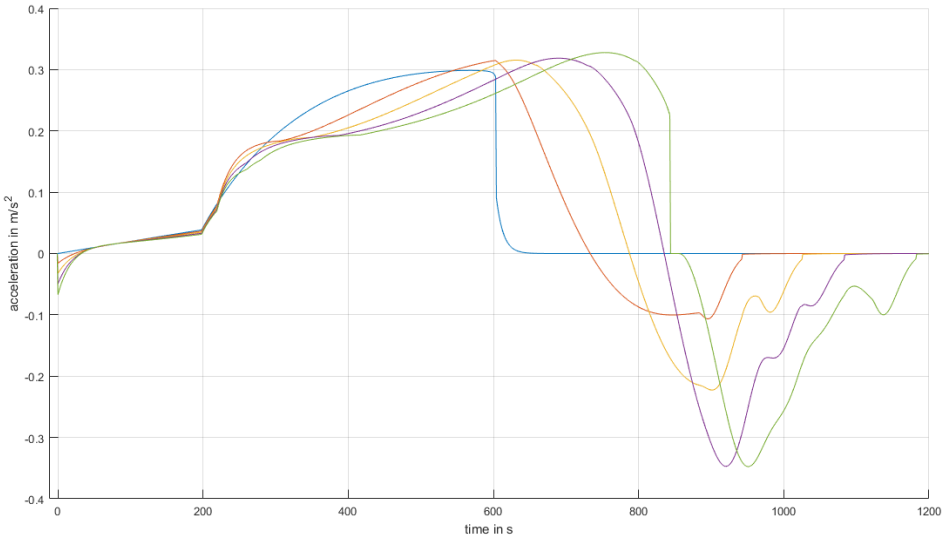

Figure 18: Acceleration graphic of a 5-car platoon with variable maximum acceleration, leader (blue), follower1(red), follower2(yellow), follower3(purple), follower4(green)

5.2.2.4 Change between controllers

As explained before, the cruise velocity and the distance reduction are made with trajectory planning functions, but after this maneuver is finished, the cars have to return to the PD controller. For this change, there is the previously described "positioning" flag that indicates which control is being used. The change from the trajectory planning to the PD controller has to be as smooth as possible. If the position or velocity error are too large, when the PD controller is switched on, an undesired peak in the acceleration to reduce this error will appear. In order to make this transition smoothly, the "positioning" flag can be set to zero when position error $|e_x| < e_{x\,max}$, the velocity error $|e_v| < e_{v\,max}$ and the acceleration error $|u_i - u_{i-1}| < e_{u\,max}$. These low errors mean that the cars keep similar velocities and accelerations, and the distance between them is the desired distance. Then, the change will be smooth from one to the other controller.

5.3 Zone 3: The changing-lane zone

In the final stretch before the second lane is obstructed, all the cars have to change to the first lane. This maneuver is executed in this third zone that goes from 300 meters before the merging point to the merging point. The 300-meter distance has been chosen because the cars can change the lane with small lateral acceleration. The implementation of the lane transition could be done using the procedure used in [4] with polynomial trajectory planning, but the aim of this project is the merging system in the longitudinal way and it was chosen to keep it as simple as possible. For this reason, the implementation in the code of this maneuver is done by subtracting a constant value to the lateral position of the vehicle until this position is equal to zero.

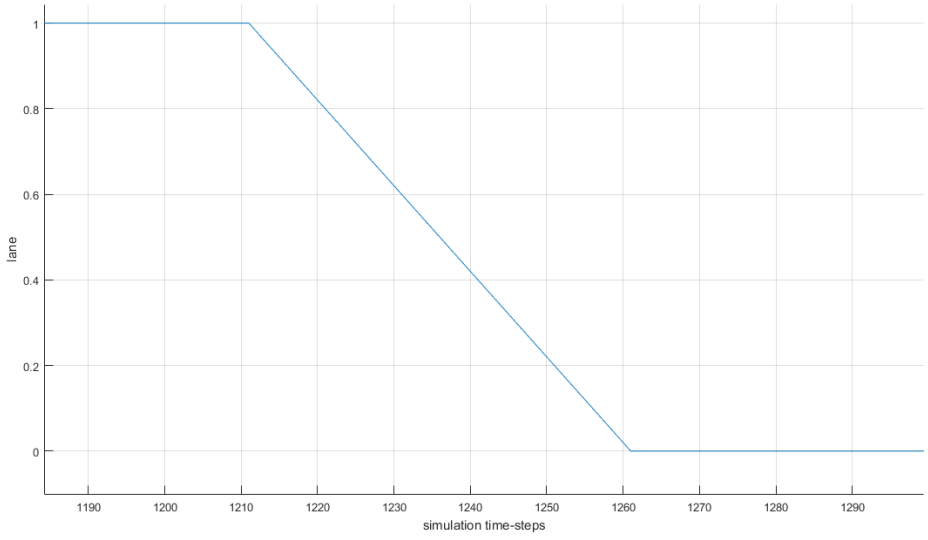

Figure 19: Changing lane process for cars in second lane

6 Implementation Progress and Code Evolution

As mentioned in the problem statement, this work has been divided in two parts. The first part was centered in the car model and the platooning. The implementation of the methods exposed in chapter 3 was done in Simulink and all the graphics of this first part of the simulation has been obtained with the Simulink Scope block. These results reference the platooning and string stability. As can be seen in Figures 3, 4 and 5 in chapter 3, the control loop provides the cars attenuation of the accelerations over the string and a collision free platooning. Moreover, in Figures 20, 21, 22 and 23, can be seen that the leader car has large negative and positive accelerations, but the follower cars with more restrictive constraints on acceleration do not collide with the predecessor car. As can be seen in figures 20 and 21, the acceleration of the leader reaches values of +/- 12 m/s^2 while the follower has a limitation on acceleration on +/- 6 m/s^2. Due to the constant time-headway policy, this sudden reduction on the velocity it also leads to a reduction of the distance between cars, this reduction of the distance indicates the controller to not brake with the same intensity as the predecessor car, helping to attenuate the large accelerations upstream.

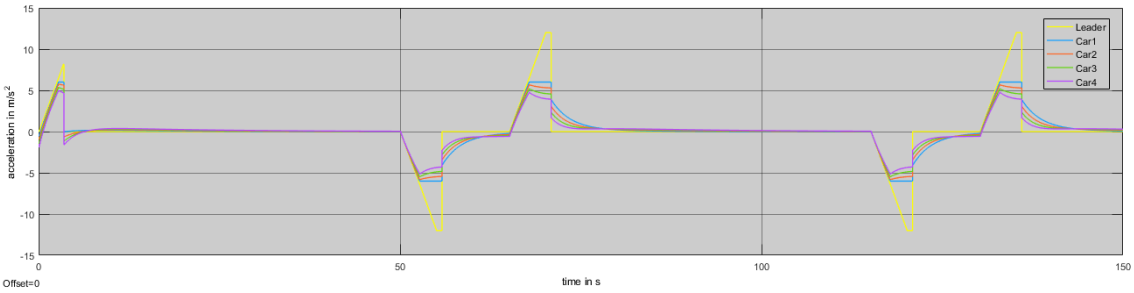

Figure 20: Acceleration using PD controller with feed-forward and constant time-headway with large acceleration on the leader

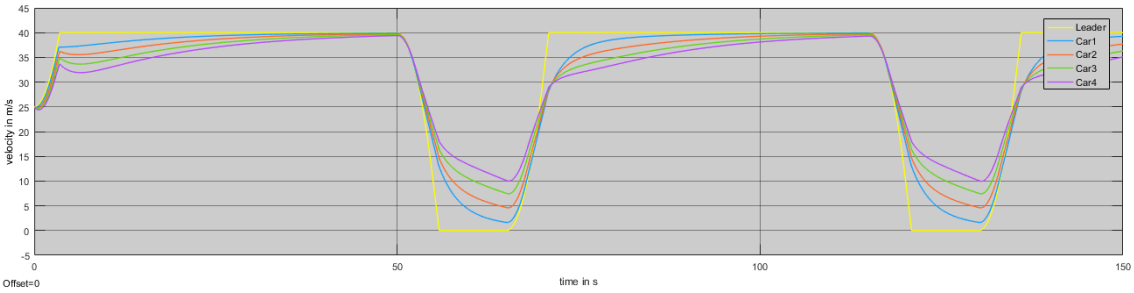

Figure 21: Velocity using PD controller with feed-forward and constant time-headway with large acceleration on the leader

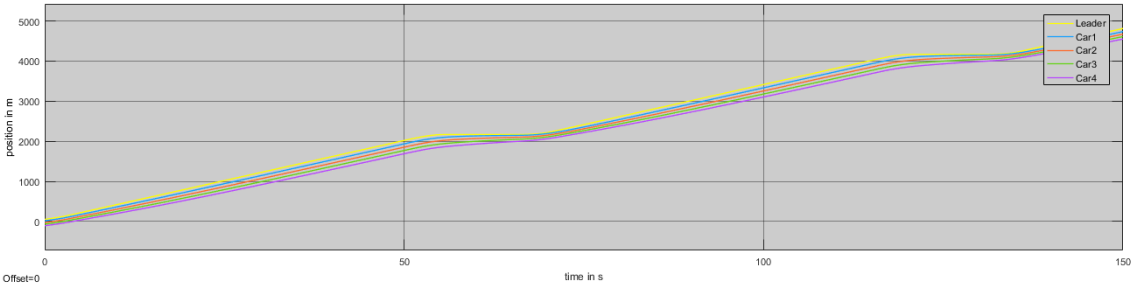

Figure 22: Position using PD controller with feed-forward and constant time-headway with large acceleration on the leader

Also, in Figures 21, 22 and 23 can be observed that when the leader is stopped (with zero velocity), the following cars do not stop. In Figure 23, it can be shown that the distance between the cars in never lower than 10 meters. That indicates that the brake has been successfully executed without any collision, even between the leader and the first follower. It can be assumed that the PD controller implemented is suitable for this application and the constant time-headway and feed-forward policy is adequate to ensure collision-free platooning and attenuation of the control signal upstream.

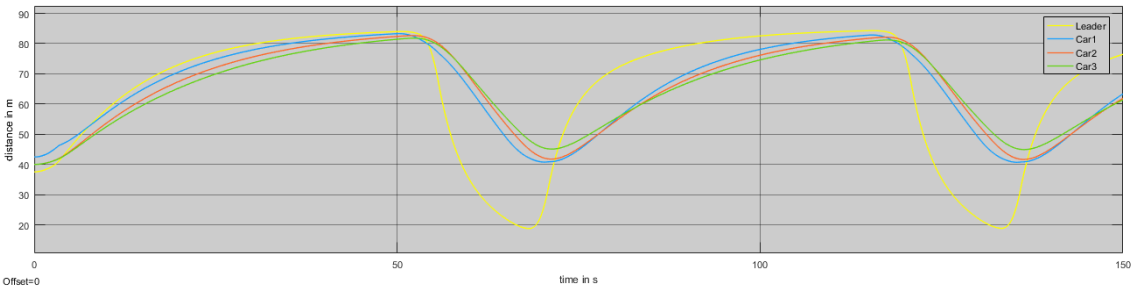

Figure 23: Distance between cars using PD controller with feed-forward and constant time-headway with large acceleration on the leader

At the beginning of the second part, the focus was set on achieving the same results with the MATLAB code as with the Simulink model. The information of the simulation had to be stored to be plotted at the end of the simulation. For this purpose, the data structure described in section 4.1.1 is important. The data matrix of every car stores the information of every step in the simulation. Then, a post process at the end of the simulation provides the same plots as obtained with the Simulink scope block.

In the first versions of the code after the successful implementation of the Simulink model, there was only implemented the PD controller when the cars were close enough to each other. If the cars were far from their predecessor, the trajectory planning for a cruise velocity was set, but the cars were experiencing big acceleration peaks when the cars reach the predecessor vehicle. As explained in section 5.2.2.1, the 4^{th} order polynomial is not a good solution for reducing the distance between cars with position constraints. Also, as can be seen in Figures 24 and 25, the change from one control to the other is not smooth

enough. Even more, the vehicles experience small oscillations in the acceleration that are uncomfortable for the passengers and may damage the vehicle.

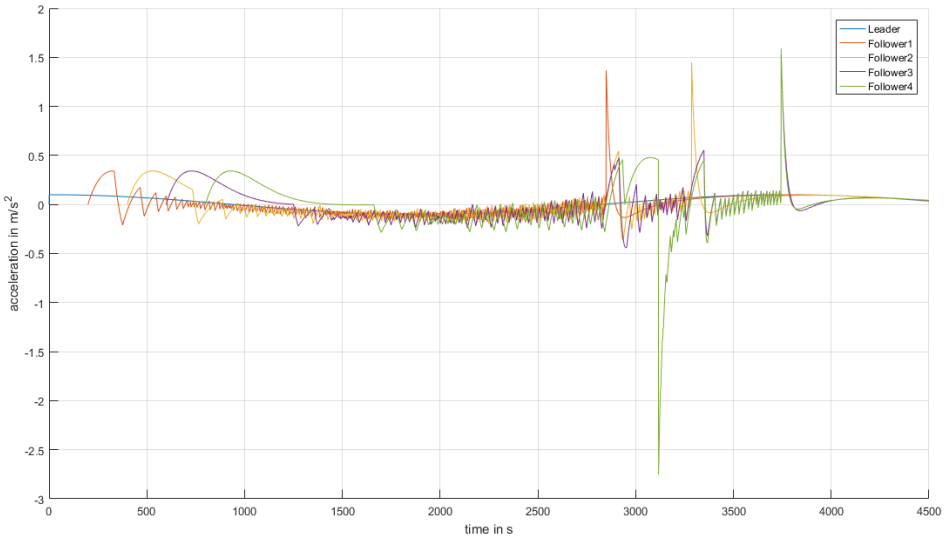

Figure 24: Platoon acceleration graphic on a first approach

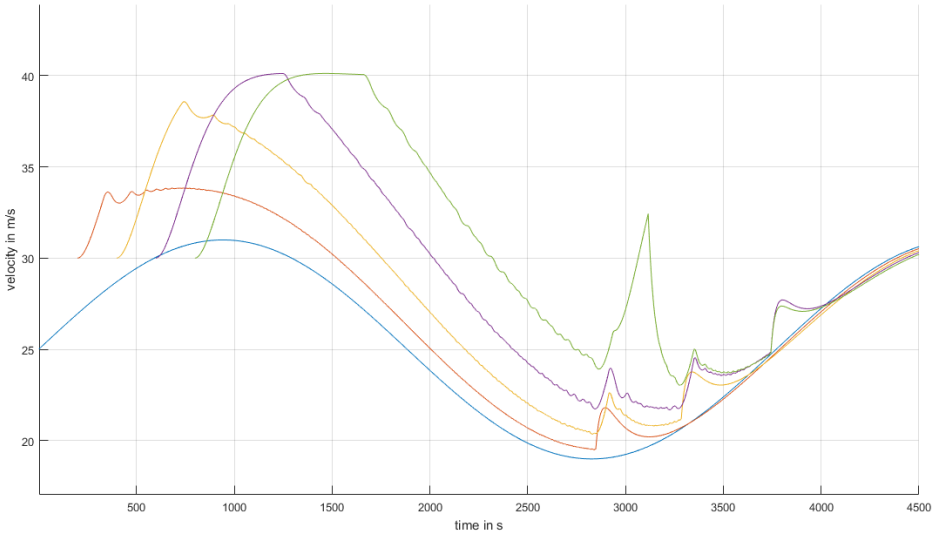

Figure 25: Platoon velocity graphic on a first approach

After the result of this experiment, it was decided to rebuild all the code. A new approach was taken, where the cars were added to the scenario in prebuilt platoons instead of lone cars that have to fuse. With this new approach can be pre-set the number of cars of the platoons in each lane and the fusing problem is removed. Once rebuilt the code and the platoons were still stable, the focus was set on the reduction of the distance between cars. In this point, the differentiation between zones was defined and introduced. The cars would maintain the platooning until the “decision-making point” where they would start reducing the

inter-vehicle distance. This procedure was implemented using the trajectory planning with 5th order polynomial as explained in sections 5.2.2.2 and 5.2.2.3 to achieve a smooth and efficient transition. This approach has been used in the next steps in this work to solve the fusing platoons feature, by changing the platoon attribute and the leader of the second platoon to a follower when this platoon is close enough to the first one to fuse (further explanation in section 5.1).

After achieving the distance reduction with smooth acceleration, the next step was to create the algorithm to decide the order of the cars. In the previous steps, all the work was developed in only one lane (platooning, distance reduction and fusing platoons), but now the second lane has also to be considered. All this new information makes more difficult to see in the plots of acceleration and velocity, but even more difficult in the position plot, what the cars are doing. In Figure 26 there is plotted the distance reduction process of platoons in both lanes. The lines of the cars cross ones with each other making more difficult the interpretation of the graphics. The solution found was to implement a new way to observe what is happening in the simulation.

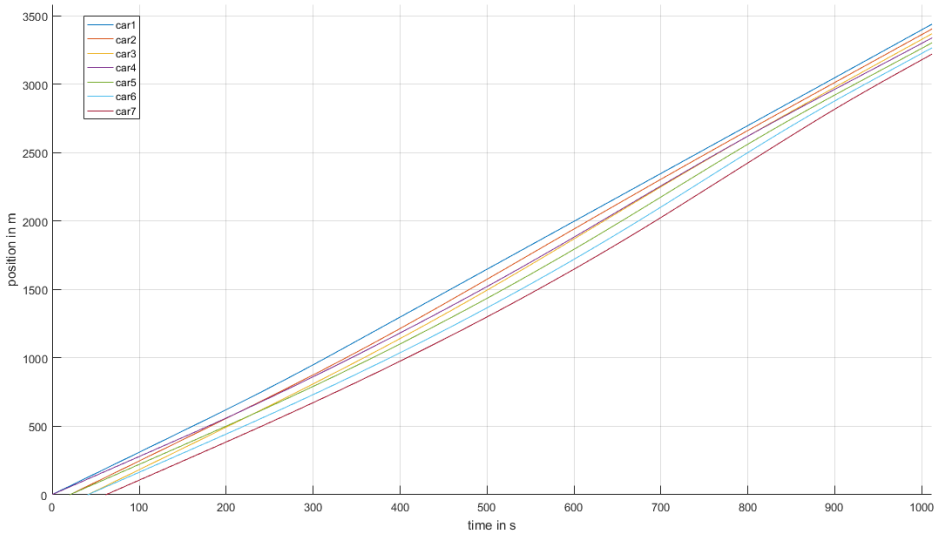

Figure 26: Position plot with cars in both lanes

6.1 Video function

Before starting to implement the merging algorithm, the creation of a function that converts the information to a video was implemented. As explained before, when the second lane is added to the plots, the lines of both platoons cross each other in the graphics and it is more difficult to see the behavior of the cars. For this reason, the idea of converting all the data to a video came up. This function is based on a code provided by A. Rupp and modified for this work. Figures 1 and 10 are examples of this function. These figures are screenshots of the generated video that have been used to explain some ideas.

This video function plots for every time step the position of all cars in the scenario. It is also printed next to each car its id, velocity and distance with the car in front. Moreover, the function can set the view on the cars with the id previously given. Before starting to process all the information, the user can enter in the function which cars wants to follow, and the function will set the center of the view on the position of the car. This video function can be set to follow more than one car each video, and after all the simulation has been processed, the information is saved in a *.avi* file that can be reproduced again.

With the implementation of the video function, previous parts of the work can be rechecked or newly implemented and seen from the perspective of one car (the same if someone was sitting on a vehicle and the other vehicles moving respectively with his car). In the first example, the distance reduction or the platoon fusing. In Figures 27, 28 and 29 is shown the fusing process of two platoons, the first one with 4 cars is slower than the second one. In the figures can be seen the smooth fusing process that also can be seen in video “fuse1.avi”.

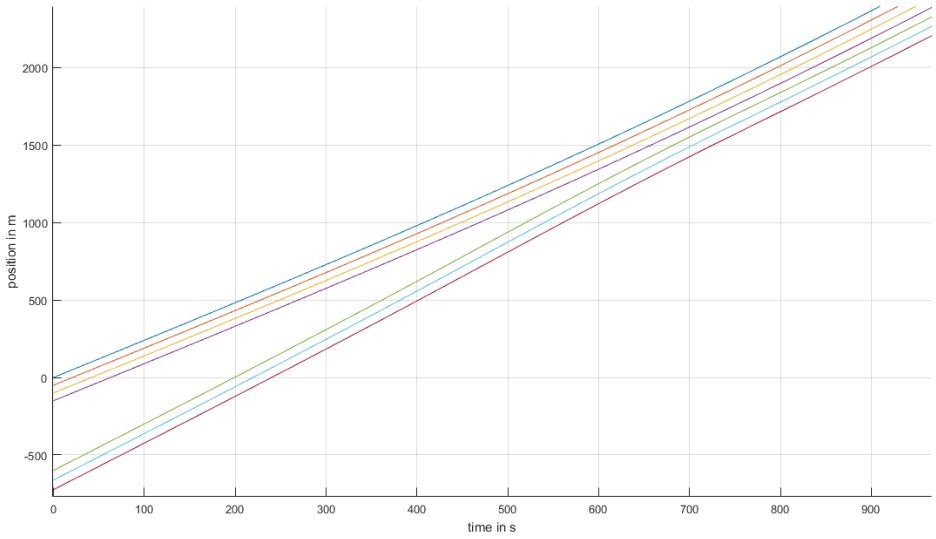

Figure 27: Position graphic of video fuse1.avi

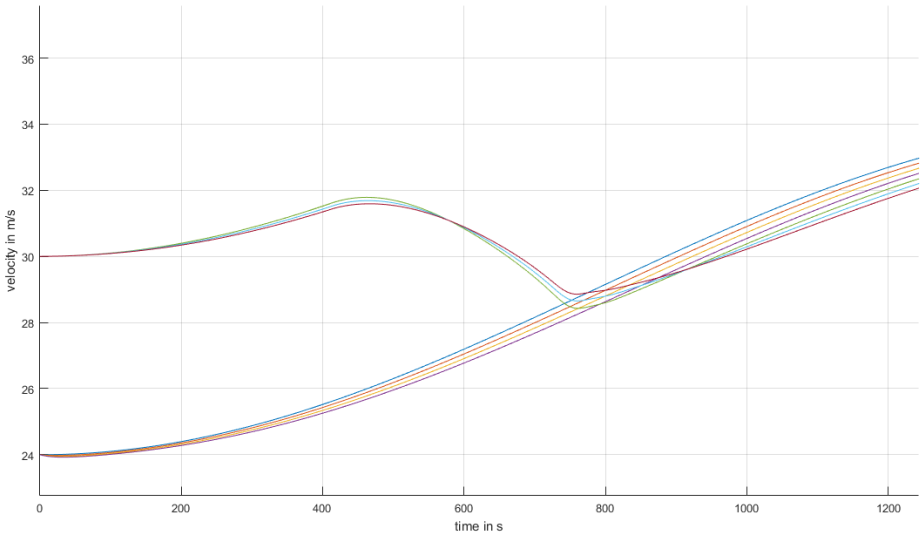

Figure 28: Velocity graphic of video fuse1.avi

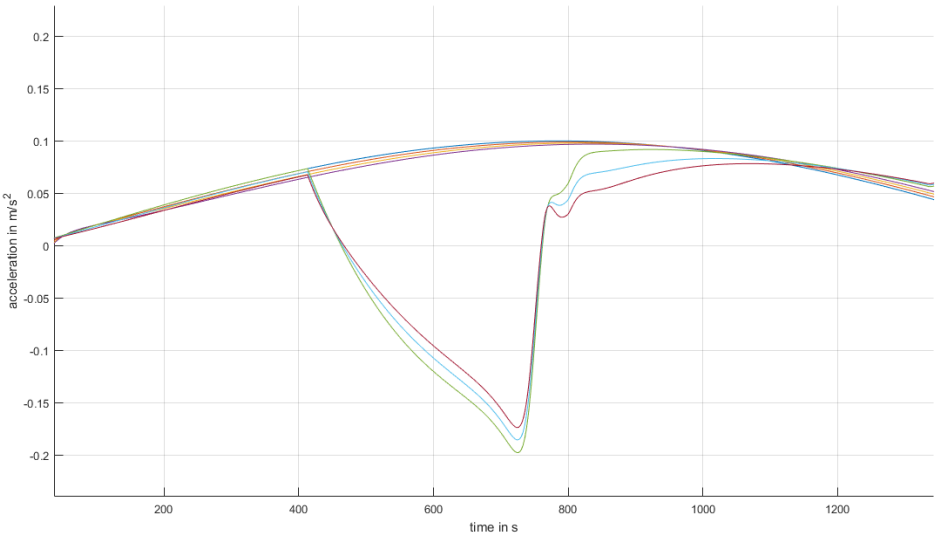

Figure 29: Acceleration graphic of video fuse1.avi

Also, in example 2, in Figures 30, 31 and 32 and video “fuse2.avi” show that the fusing process and the distance reduction can be executed simultaneously. The difference between the example 1 and this example can be seen in the final position of the cars. In example 2, the lines in the position graphic are closer. This means that the distance between them is smaller because while the cars were fusing, simultaneously, were reducing the gap between them.

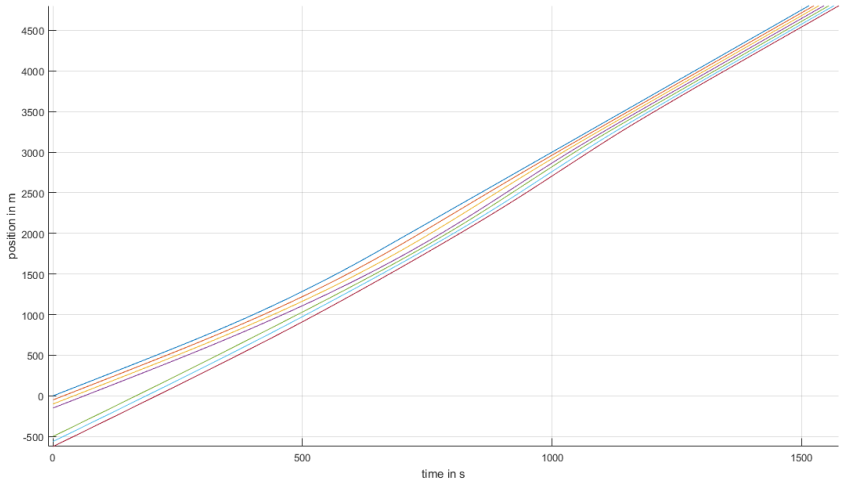

Figure 30: Position graphic of video fuse2.avi

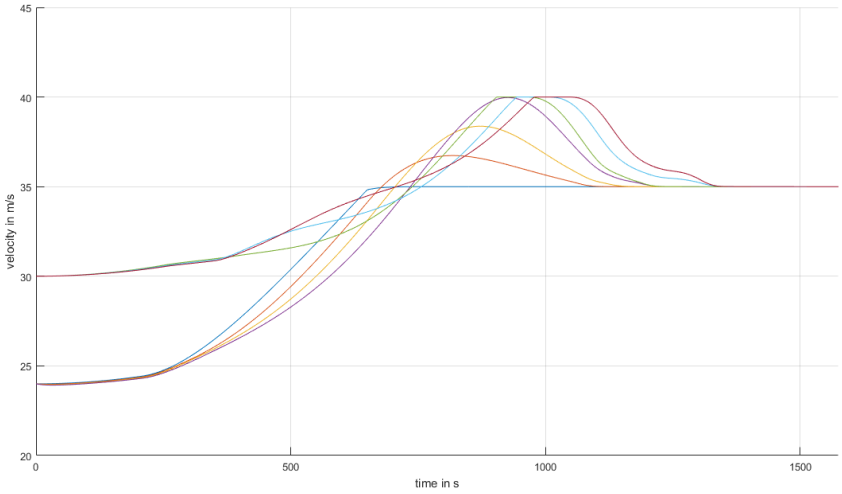

Figure 31: Velocity graphic of video fuse2.avi

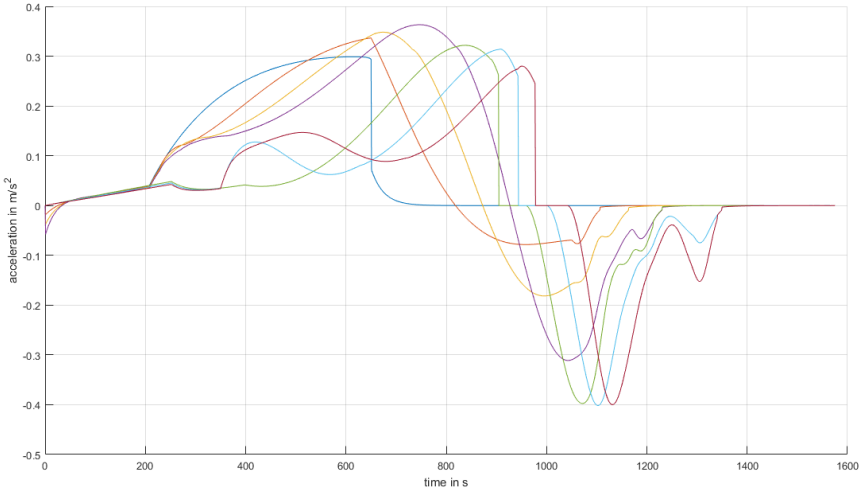

Figure 32: Acceleration graphic of video fuse2.avi

The last step of this work was implementing the merging algorithm based on the decision-making algorithm presented in section 5.2.1. In the first approach, the cars were checking all the time if they should change the order or not. In the “video1.avi” can be seen the first successful merging with 3 cars in the main lane and only one in the first lane. It can be seen in this video that the changing lane feature was still unimplemented, but the final position is correct. After this first success, the changing lane feature was added, and a new experiment was set up with 5 cars in each lane. This new trial can be seen in “video2.avi”. In this second video can be seen the two thing that have been tried to prevent. The first one is the collision of vehicles, this is the most serious one. The second one is the inefficient acceleration and brake (explained in section 5.2.2.3 and in Figure 16. When a car is accelerating, braking and accelerating again is losing the energy in the unnecessary brake caused by a bad trajectory planification). From 00:11 to 00:32 in “video2.avi” can be seen this behavior of cars with id2 and id3. They accelerate to reduce the distance with the car in front, but later they have to brake to enlarge the gap for the second lane cars, and then accelerate again to reach the acceleration of the leader. Also, from 00:42 to 00:50 the two last cars of each platoon collide with each other. After this result, was decided that this is not a good approach.

In the second approach, was chosen that the decision making was taken in only one point on the highway (the decision-making point, at the beginning of zone 2). Where the cars have to decide the new order in the overall platoon. With this new approach the complexity of the data storage system was evidenced when the cars had to change from one platoon to another. Before this point the structs of the cars of each platoon were saved in a platoon struct, and all this platoon structs in the overall struct of cars, see Figure 33. The code had to be rewritten and the data structure was modified until reaching the form seen in the section 4.1.1.

This code revision helped to improve the efficiency and a clearer structure, but also accomplish the free and efficient merging system. As can be observed in “video3.avi”, two platoons of four cars each are merged successfully. Although the velocities are equal in both platoons, they have zero acceleration and the platoon in the second lane is slightly behind the one in the main lane, the merging algorithm works perfectly. After proving that this approach is suitable (this was the easy case, it had to be improved), new simulations were set. In video “video4.avi” two platoons of 8 cars in each lane are introduced in the scenario at the same moment in the same position, but with different velocity. This velocity difference made the cars of the second lane take later positions in the overall platoon after the decision-making point. As can be seen from 00:10 to 00:25, the increment of velocity of the cars id0, id1 and id2 is higher than the car id8. This is because the car with id8 after the decision-making point is order 4 in the overall platoon and has to wait until the other cars have overtaken it to accelerate. The same situation happens with id3, the distance between id3 and its predecessors is big, but all the cars are accelerating, and his polynomial trajectory planning do

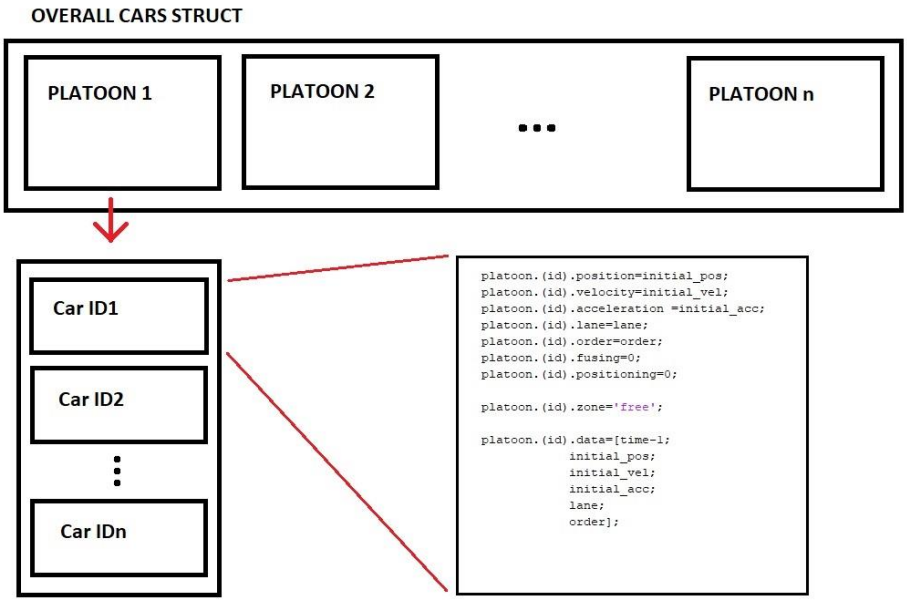

Figure 33: Schematic of the data structure in previous versions

not allow him to accelerate too much. Also, the fact that its velocity is similar to the velocity of its predecessor (id8) makes the polynomial less aggressive.

Finally, in video “video5.avi” can be seen an improved version of the code used in “video4.avi”. This video is the same used to take Figure 10. In this final version and more specifically in this video can be observed that the merging sequence do not have to be always like a zip of the two platoons. As the video shows, the new order in the overall platoon is AABABBABAB. In this version there is an improvement the problem of the large spaces between cars when they have to be overtaken. Although car id2 is giving to much space with id1 and id5, all these cars are accelerating almost at the same ratio. Also, the car with id3 has to enlarge the gap more than usual to let two cars of the second lane merge in front of it but executes a smooth transition.

7 Results

With the final version of the code, the next simulations have been done. In this final version, the number of cars in the platoons are randomly set, and each simulation could be a different number in each lane. Also, to prevent that the leaders of both platoons start at the same point, the cars of the second lane do not always start on the initial coordinate of position = 0. These two features add randomness to every simulation, providing every time new setups.

In the first example, two platoons, each of which has 4 cars, are merging alternatively. The platoons in the first lane are slightly faster and they take the lead in the overall platoon. The expected result can be seen in Figures 34, 35 and 36 and also in video “Final1.avi”. Also, can be observed that the leader of the platoon of the second lane take the 3rd position due to the speed difference, for this reason, it is not a simple zipper merging sequence. These results indicate that it is possible to merge smoothly using the merging control algorithms presented here.

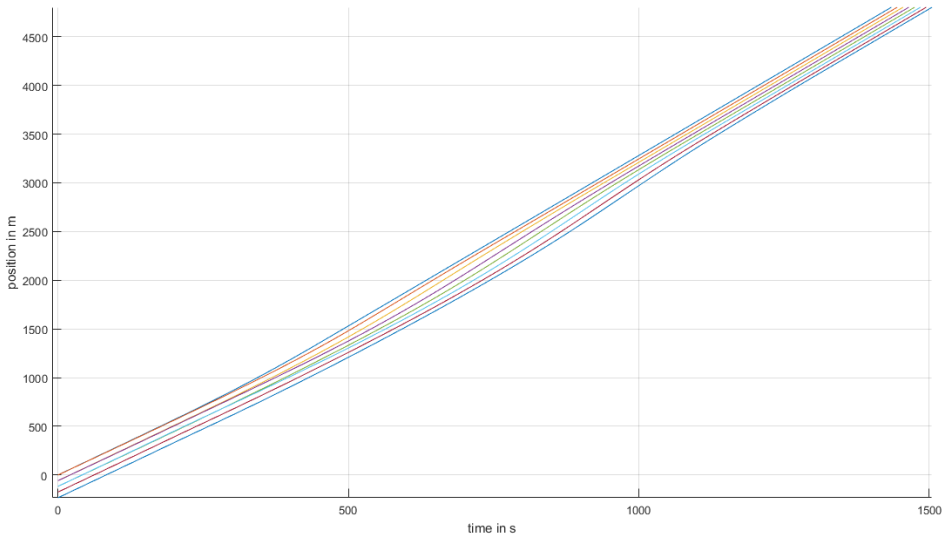

Figure 34: Position plot of video final1.avi

In the second example recorded in “final2.avi”, shows that the merging sequence it is not a pure zipper merging sequence either, where two cars of the second lane merge consecutively. Also, as can be seen in “final3.avi” the leader of the platoon of the first lane does not have to be always the leader of the overall platoon. In this third video shows that the first two cars of the platoon of the second lane are faster than the leader of the first lane and because of the algorithm, it takes the 3rd position in the overall platoon.

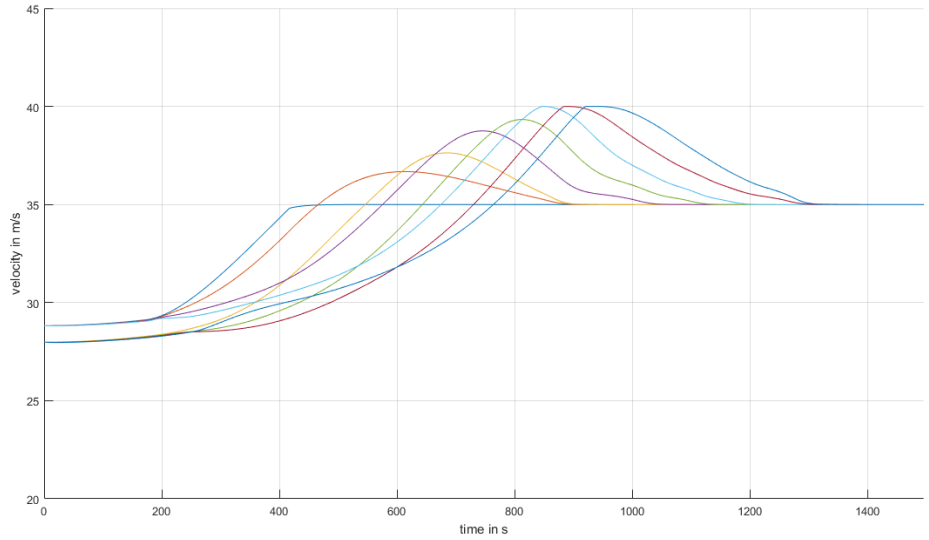

Figure 35: Velocity plot of video final1.avi

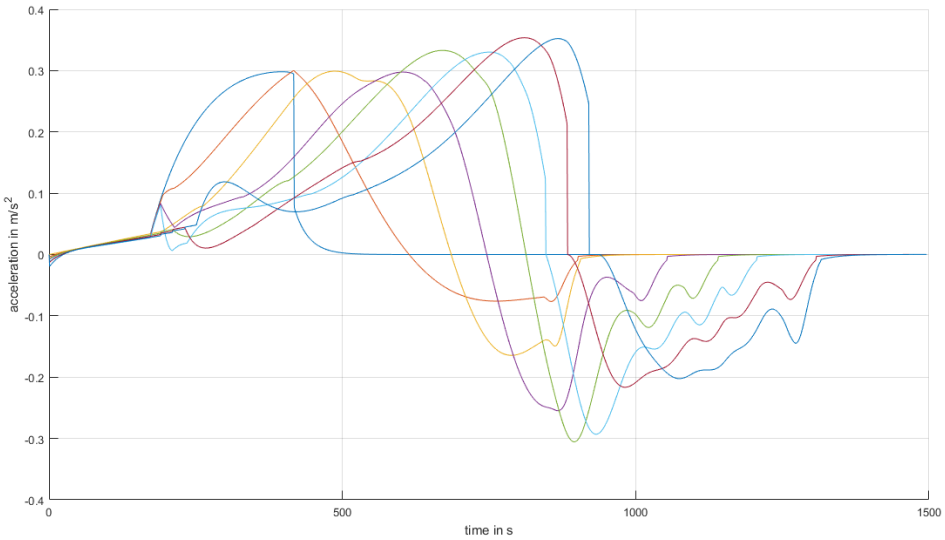

Figure 36: Acceleration plot of video final1.avi

8 Conclusions and Future Work

In this project, the problem of reducing a lane on a highway is discussed. These situations can cause traffic jams or difficulties in the vehicle's circulation. This work is based on a two-lane highway with a lane reduction, and cars have to merge autonomously and maintain a continuous flow. For this reason, a solution based on the zipper principle was proposed. The implementation of this solution started from the basic model of the car and the design of the controller until the complete implementation of the fusion algorithm.

In this work, only the longitudinal control on a straight road is considered. In future versions, the lateral control and the layout of the road can also be taken into account. The design of a decentralized algorithm for controlling the merging of autonomous vehicles was one of the main points. In addition, the string stability is an essential requirement for the design of a following control system with a short distance between the cars. The algorithm presented guarantees a merging of the vehicles without collisions and a significant increase of the road throughput. In addition, in the case of heavy vehicles, a decrease in fuel consumption and emissions can be expected.

The method presented can give a smooth trajectory for the merging sequence of vehicles, avoid the accelerations jumps when the distance constraint changes or a demand for speed when the predecessor car accelerates. Also, a kind of merging control based on the virtual platoon concept is introduced (not all the cars of the overall platoon are always in the same lane), and this approach can be further investigated in future updates. The implementation of this approach is not a finished version, there are still necessary modifications and improvements, but it is a solid base from which to continue to expand and upgrade the system. Next steps of this algorithm could include features of predictive and adaptive control. Combining this approaches with the algorithm implemented in this project could increase the efficiency and the robustness of the merging order decision.

In addition, the current version of the code could be improved. One of the next steps could be the transition from the struct model to an object-oriented approach. All cars can be modeled into classes with built-in methods. This new approach would simplify the structure of the code and its implementation. Also, finishing the implementation of the continuous flow simulation to test the behavior of the algorithm on a standard highway, and the simulation of the proposed approach in SUMO to check the reliability of the implemented method in a specialized environment could bring new ideas or show deficiencies in the presented approach. Exploring the possibility of improving the system from 2-to-1 to 3-to-2 lane reduction could be a great challenge. Due to the fact that nowadays highways usually have 3 lanes each direction, this new algorithm could be a starting point for a new merging sequence method that could be applied to most European highways.

9 References

[1]. Newton, “Philosophiæ Naturalis Principia Mathematica”. July 1687

[2]. MATLAB documentation, “Discrete-Time Integrator block”, https://de.mathworks.com/help/simulink/slref/discretetimeintegrator.html. accessed: Oct 2017

[3]. MATLAB documentation, “lqrd MATLAB function”, https://de.mathworks.com/help/control/ref/lqrd.html. accessed: Oct 2017

[4]. A. Rupp, “Trajectory planning and Formation Control for automated driving on highways”, PhD thesis, Graz University of technology, 2018

[5]. F. Morbidi, P. Colaneri, T. Stanger, “Decentralized optimal control of a car platoon with guaranteed string stability”. July 2013

[6]. X. Yu, G. Guo, J. Gong, “Cooperative control for a platoon of vehicles with leader-following communication strategy”. Oct. 2017

[7]. W. B. Dunbar, D. S. Caveney, "Distributed receding horizon control of vehicle platoons: stability and string stability", Mar. 2012.

[8]. T. Bellemans, B. De Schutter, B. De Moor, “An improved first-order macroscopic flow model for highway traffic simulation”. June 2001

[9]. J. Ploeg, N. van de Wouw, H. Nijmeijer, “*Lp* String Stability of Cascaded Systems: Application to Vehicle Platooning”. March 2014

[10]. R. Olfati-Saber, R. M. Murray, “Consensus Problems in Networks of Agents with Switching Topology and Time-Delays”. Sept 2004

[11]. W. REN, R. W. BEARD, E. M. ATKINS, “Information Consensus in Multivehicle Cooperative Control”. Apr 2007

[12]. J. Ploeg, B. T. M. Scheepers, E. van Nunen, N. van de Wouw, H. Nijmeijer,” Design and Experimental Evaluation of Cooperative Adaptive Cruise Control”. Oct 2011

[13]. S. Santini, A. Salvi, A. S. Valente, A. Pescapè M. Segata, R. Lo Cigno, “A Consensus-based Approach for Platooning with Inter-Vehicular Communications”. Apr 2015

[14]. A. Morales, H. Nijmeijer, “Merging Strategy for Vehicles by Applying Cooperative Tracking Control”. May 2016

[15]. C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. Hoang Bengtsson, J. DidoffThe, “Grand Cooperative Driving Challenge 2016: Boosting the Introduct ion of Cooperative Automated Vehicles”. June 2016

[16]. A. Mosebach, S. Röchner, J. Lunze, “Merging control of cooperative vehicles”. June 2016

[17]. A. Uno, T. Sakaguchi, S. Tsugawa, “A Merging Control Algorithm based on Inter-Vehicle Communication”. Oct 1999

[18]. C. Guo, C. Sentouh, J. Popieul, B. Soualmi, J. Haué, “Shared Control Framework Applied for VehicleLongitudinal Control in Highway Merging Scenarios”. Oct 2015

[19]. L. Wu, X. Chen, “The Automated Vehicle Merging Based on Virtual Platoon”. Sept 2015
[20]. M. Zabat, N. Stabile, S. Frascaroll, F. Browand, “The Aerodynamic Performance of Platoons”, July 2011